GOPAS
IT ŠKOLIACE STREDISKO

.NET Memory Internals

ShowIT 2016
& Security Day

Ing. Bc. Robert Haken
MVP Development | MCT
haken@havit.cz | @RobertHaken

Agenda

Architektura paměti .NET aplikací

Stack × Managed Heap

Hodnotové × referenční datové typy

Volání metod a předávání parametrů

Garbage Collection

DEMO

Windows Debugger a inspekce paměti

01-StackHeap, x64 DMP

Architektura paměti .NET apps

Instrukční paměť

Zásobník (Stack)

Halda (Heap)

- Native Heap (Unmanaged Resources)
- Managed Heap (.NET, Garbage Collector)

Zásobník (Stack)

per thread

fixed size (konfigurovatelný, default 1 MB)

LIFO – Last In, First Out

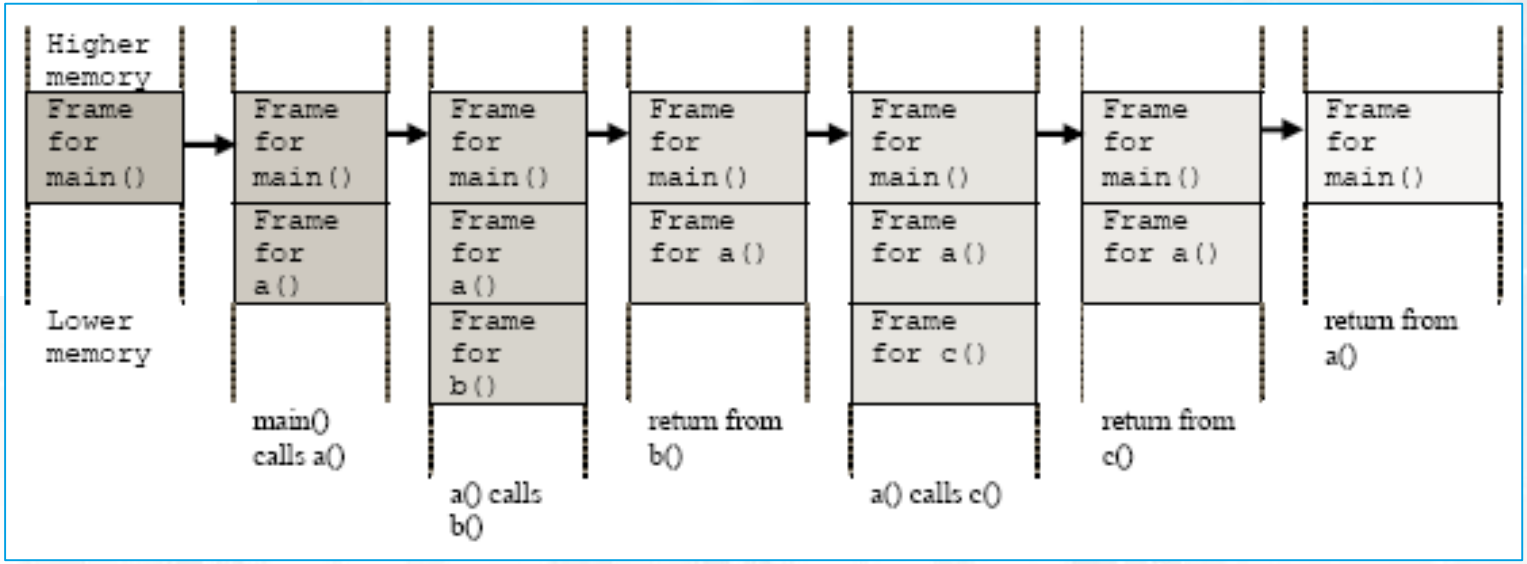
Activation Records (Stack Frames)

- parametry (argumenty)
- návratová adresa, registry CPU, ...
- lokální proměnné

Stack a volání metod

```
void main()
{
    a();
}

void a()
{
    b();
    c();
}
```



DEMO

Stack Frames (01-StackHeap, X64)

Datové typy

Hodnotové

- in-place hodnota
- primitivní typy
- `Int16/32/64`, `Byte`, `Boolean`, `Double`, ...
- char, Decimal
- **`struct`** - `DateTime`, `Nullable<T>`, vlastní, ...

Referenční

- „hodnotou“ je odkaz na instanci (tj. adresa)
- `class` (vč. `Object`)
- pole – `Array`, `type[]`
- `string`
- na jednu instanci může ukazovat více referencí

= operace přiřazení

zkopírování paměťové buňky

- hodnotový typ → zkopírování vlastní hodnoty
- referenční typ → zkopírování odkazu na instanci

DEMO

Hodnotové/referenční typy, přiřazování
(02-ValueAndReferenceTypes, x86)

Předávání parametrů do metod

předání parametru „hodnotou“ (default)

→ zkopírování paměťové buňky

předání parametru „referencí“ (`ref`, `out`)

→ odkaz na zdrojovou paměťovou buňku (stack)

návratová hodnota metody

→ zkopírování paměťové buňky (~ přiřazení)

DEMO

Předávání parametrů do metod (03-CallingMethods, x86)

Halda (Managed Heap)

dynamická velikost

alokace při vytváření instancí (new)

pod správou Garbage Collectoru

Garbage Collection

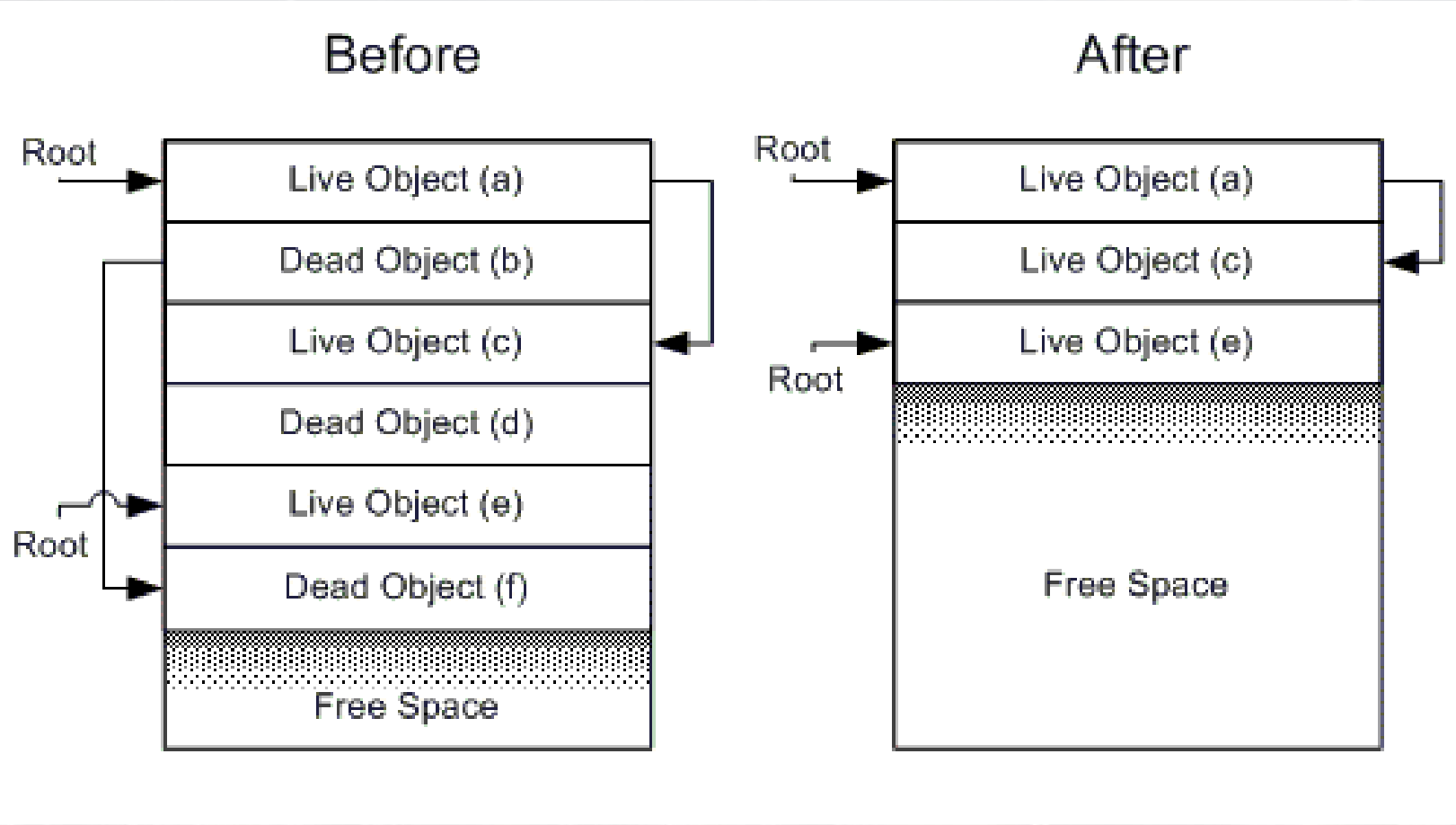
Garbage Collector ~ 1959 LISP

.NET Garbage Collection

- překročení thresholdu generace při alokaci
- system-wide memory pressure
- GC.Collect() API (never-ever!)

sesypání, update referencí

Garbage Collection - sesypání



DEMO

Garbage Collection
(02-SimpleGarbageCollection, x86)

Roots

kořeny grafu objektů pro zjištění dosažitelnosti

- zásobník (lokální proměnné, parametry metod)
- GCHandles
  - globální statické fieldy
  - pinned objects
  - ...
- F-reachable queue

(domácí úkol: WeakReference)

DEMO

GC Roots (03-GCRoots, x86)

Generations

výkonová optimalizace

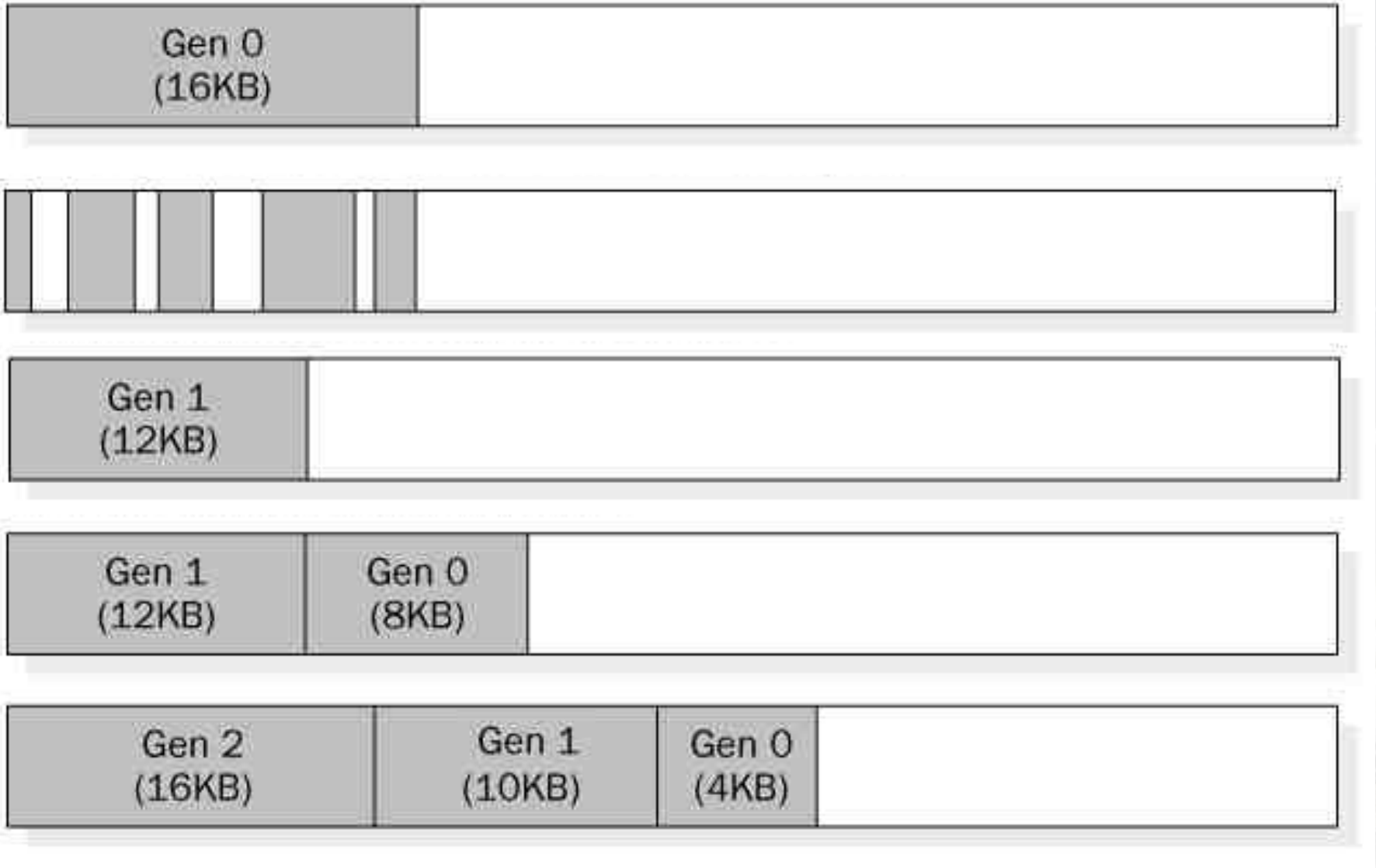
soustředí se objekty s krátkou životností

Gen 0 – nové, Gen 1 & 2 – přežily 1/více-krát

Gen 0 + 1 = ephemeral segment („fixed“ size)

Gen 2 = variable size

Generations



Finalization

explicitní úklid unmanaged zdrojů

Finalize() ~ C# destructor

Finalization Queue => vždy Gen 1

F-reachable Queue

Finalization Thread

IDisposable, ResourceWrapper pattern

Large Object Heap

výkonová optimalizace

objekty větší než 85 000 bytů (default)

součást Gen 2 collection

nesetřásá se (NET 4.5.1+ lze jednorázově)

Free List

Verze Garbage Collectoru

Workstation (lag) vs. server (throughput)

```
configuration/runtime/gcServer enabled="true|false"
```

Background GC (NET4 wks only, NET4.5 svr)

```
configuration/runtime/gcConcurrent enabled="true|false"
```

Objekt >2GB (x64, NET4.5)

LOH Free Lists Optimization (NET4.5)

LOH Heap Balancing (NET4.5)

LOH Compaction, explicit (NET4.5.1)

www.showit.sk

www.gopas.sk